/// original design by GrayMatter Tutorials

<https://www.youtube.com/watch?v=M1jQNotyVfg&list=PL1talji0WeRPz_NJJGOcEgibOJwbttmc9>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class QUESTSYSTEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

using ItemSystem;

using ActorSystem;

using DialogueSystem;

namespace QuestSystem {

public class Quest : IIdentity , IClonable<Quest>, IEditable {

public static event SFEventHandler<Quest> Completed;

public static event SFEventHandler<Quest> Began;

private string \_name;

private string \_description;

private QuestType \_type;

private QuestState \_status;

private bool \_isOptional;

private string \_actorName;

private string \_itemName;

private int \_quantityNeeded;

private int \_quantityRemaining;

#region Getters & Setters

public string name { get { return \_name; } set { \_name = value; } }

public string decription { get { return \_description; } set { \_description = value; } }

public QuestType type { get { return \_questType; } set { \_questType = value; } }

public QuestStatus status { get { return \_questStatus; } set { \_questStatus = value; } }

public bool isOptional { get { return \_isOptional; } set { \_ isOption = \_value; } }

public string actorName { get { return \_actorName; } set { \_actorName = value; } }

public string itemName { get { return \_itemName; } set { \_itemName = value; } }

public int quantityNeeded { get { return \_quantityNeeded; } set { \_quantityNeeded = value; } }

public int quantityRemaining { get { return \_quantityRemaining; } set {\_quantityRemaining = value; } }

#endregion

#region Constructors

public Quest() : this (string.Empty, string.Empty, QuestType.None, string.Empty, string.Empty, 0) { }

public Quest(string name, string desc, QuestType type, string actorName, string itemName, int quantityNeeded, bool isOptional = false) {

\_name = name;

\_description = desc;

\_type = type;

\_status = QuestStatus.Inactive;

\_actorName = actorName;

\_itemName = itemName;

\_quantityNeeded = quantityNeeded;

\_quantityRemaining = \_quantityNeeded;

\_isOptional = isOptional;

}

public Quest Clone() {

return (Quest)MemberwiseClone();

}

#endregion

private void AddEventListeners() {

switch(\_type) {

case QuestType.Collection:

Item.Collected += Item\_Collected;

break;

case QuestType.Delivery:

Actor.Delivered += Actor\_DeliveredItem;

break;

case QuestType.Elimination:

Actor.Eliminated += Actor\_Eliminated;

break;

case QuestType.Interaction:

Actor.Interacted += Actor\_Interacted;

break;

}

}

private void RemoveEventListeners() {

switch(\_type) {

case QuestType.Collection:

Item.Collected -= Item\_Collected;

break;

case QuestType.Delivery:

Actor.Delivered -= Actor\_DeliveredItem;

break;

case QuestType.Elimination:

Actor.Eliminated -= Actor\_Eliminated;

break;

case QuestType.Interaction:

Actor.Interacted -= Actor\_Interacted;

break;

}

}

private void Item\_Collected(Item sender) {

if (\_status = QuestStatus.Active && sender.name == \_itemName ) {

\_quantityRemaing--;

if (\_quantityRemaining <=0 )

\_status = QuestStatus.Pending;

}

}

private void Actor\_Delivered(Actor sender, Item args) {

if (\_status = QuestStatus.Active && sender.name == \_actorName && args.name == \_itemName ) {

\_quantityRemaing--;

if (\_quantityRemaining <=0)

\_status = QuestStatus.Pending;

}

}

private void Actor\_Eliminated(Actor sender) {

if (\_status = QuestStatus.Active && sender.name == \_actorName ) {

\_quantityRemaing--;

if (\_quantityRemaining <=0)

\_status = QuestStatus.Pending;

}

}

private void Actor\_Interacted(Actor sender) {

if (\_status = QuestStatus.Active && sender.name == \_actorName )

\_status = QuestStatus.Pending;

}

public void Complete() {

if (\_status == QuestState.Pending) {

ForceComplete();

}

}

public void ForceComplete() {

\_status = QuestStatus.Complete;

RemoveEventListeners();

if (Completed != null)

Completed(this);

}

public void Begin() {

if (\_status == QuestState.Inactive) {

ForceBegin ();

}

}

public void ForceBegin() {

\_status = QuestStatus.Active;

AddEventListeners();

if (Completed != null)

Began(this);

}

public void RevertFromDatabase() { }

public void OnInspectorGUI() {

}

public void OnEditorGUI() {

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class ACTORSYSTEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using ItemSystem;

using QuestSystem;

namespace ActorSystem {

[System.Serializable]

public class Actor : IIdentity , IClonable<Actor>, IEditable {

public static SFEventHandler<Actor, Item> DeliveredItem;

public static SFEventHandler<Actor> Eliminated;

public static SFEventHandler<Actor> Interacted;

private string \_name;

private string \_nickName;

#region Getters & Setters

public string Name { get { return \_name; } set { \_name = value; } }

public string NickName { get { return \_nickName; } set { \_nickName = value; } }

#endregion

#region Constructors

public Actor() {

\_name = “New Actor”;

}

public Actor Clone(){

return (Actor)MemberwiseClone();

}

#endregion

public void RevertFromDatabase() {

}

public void OnInspectorGUI() {

EditorGUILayout.BeginHorizontal();

if (GUILayout.Button(“Open”, EditorStyles.miniButtonLeft)) { }

if (GUILayout.Button(“Revert”, EditorStyles.miniButtonMid)) { }

if (GUILayout.Button(“Apply”, EditorStyles.miniButtonRight)) { }

EditorGUILayout.EndHorizontal();

EditorGUILayout. LabelField (“Name: ”, \_name);

EditorGUILayout.TextField (“Nickname: ”, \_nickName);

if (GUILayout.Button(“Sync with Database”)) {

//Use name to reload info from database

}

}

public void OnEditorGUI() {

EditorGUILayout. TextField (“Name: ”, \_name);

EditorGUILayout.TextField (“Nickname: ”, \_nickName);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class ITEMSYSTEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

namespace ItemSystem {

public class Item : IIdentity , IClonable<Item>, IEditable {

public static event SFEventHandler<Item> Collected;

private string \_name;

private string \_description;

private Sprite \_icon;

private int \_value;

private bool \_isLore;

private bool \_isTradeable;

private bool \_isMagic;

private ItemRarity \_rarity;

#region Getters & Setters

public string Name { get { return \_name; } set {\_name = value; } }

public string Description { get { return \_ description; } set {\_ description = value; } }

public Sprite Icon { get { return \_ icon; } set {\_ icon = value; } }

public int Value { get { return \_ value; } set {\_ value = value; } }

public bool IsLore { get { return \_isLore; } set { \_isLore = value; } }

public bool IsTradeable { get { return \_isTradeable; } set { \_isTradeable = value; } }

public bool IsMagic { get { return \_isMagic; } set { \_isMagic = value; } }

public ItemRarity { get { return \_rarity; } set { \_rarity = value; } }

#endregion

#region Constructors

public Item() : this (string.Empty, string.Empty, null, 0, false, true, false, ItemRarity.Common) { }

public Item(string name, string desc, int value, bool lore, bool trade, bool magic, ItemRarity rare) {

\_name = name;

\_description = desc;

\_value = value;

\_isLore = lore;

\_isTradeable = trade;

\_isMagic = magic;

\_rarity = rare;

}

public Item Clone() {

return (Item)MemberwiseClone();

}

#endregion

public static void CollectItem(Item item) {

if (Collected !=null)

Collected(item);

}

public void RevertFromDatabase() { }

public void OnInspectorGUI() { }

public void OnEditorGUI() { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Equipment \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem;

public class Equipment: Item, Prefab {

private int \_armorRating;

private int \_durability;

private int \_maxDurability;

private EquipmentSlot \_slot;

#region Getters & Setters

public int ArmorRating { get { return \_armorRating; } set { \_armorRating = value; } }

public int Durability { get { return \_durability; } set { \_durability = value; } }

public int MaxDurability { get { return \_maxDurability; } set { \_maxDurability = value; } }

public EquipmentSlot Slot { get { return \_slot; } set { \_slot = value; } }

#endregion

#region CONSTRUCTORS

public Equipment() : this ( 0, 0, 0 , EquipmentSlot.none ) {}

public Equipment (string name, string desc, int cost, bool lore, bool magic, bool trade, ItemRarity rare, int dur, int maxDur, EquipmentSlot slot, GameObject prefab, int armorRate){

\_name = name;

\_description = desc;

\_cost = cost;

\_isLore = lore;

\_isMagic = magic;

\_isTradeable = trade;

\_rarity = rare;

\_durability = dur;

\_maxDurability = maxDur;

\_slot = slot;

\_prefab = prefab;

\_armorRating = armorRate;

}

#endregion

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Weapon \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public class Weapon : Item, Prefab {

private float \_minDam, \_maxDam;

private float \_meleeRange, \_missileRange;

private WeaponType \_type;

#region SETTERS & GETTERS

public float MinDam { get { return \_minDam; } set { \_minDam = value; } }

public float MaxDam { get { return \_maxDam; } set { \_maxDam = value; } }

public float MeleeRange { get { return \_meleeRange; } set { \_meleeRange = value; } }

public float MissileRange { get { return \_missileRange; } set { \_missileRange = value; } }

public WeaponType Type { get { return \_type; } set { \_type = value; } }

#endregion

#CONSTRUCTORS

public Weapon() : this ( 0, 0, 0, 0, WeaponType.None ) {}

public Weapon(string name, string desc, int cost, bool lore, bool magic, bool trade, ItemRarity rare, float min, float max, float melee, float missile, WeaponType type, GameObject prefab) {

\_name = name;

\_description = desc;

\_cost = cost;

\_isLore = lore;

\_isMagic = magic;

\_isTradeable = trade;

\_rarity = rare;

\_minDamage = min;

\_maxDamage = max;

\_meleeRange = melee;

\_missileRange = missile;

\_type = type;

\_prefab = prefab;

}

#endregion

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class JEWELRY SYSTEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public class Jewelry : Item {

private List<Buff> \_buffs;

private JewelrySlot \_slot;

#region SETTERS & GETTERS

public List<Buff> Buffs {

get { for (int i = 0;i <\_buffs.Count; i++)

return \_buffs[i]; }

set { \_buffs.Add(value); } }

public JewelrySlot Slot { get {return \_slot; } set { \_slot = value; } }

#endregion

#region CONSTRUCTORS

public Jewelry() : this ( JewelrySlot slot, List<Buff> buffs){}

public Jewelry(string name, string desc, int cost, bool lore, bool magic, bool trade, ItemRarity rare, List<Buff> buffs, JewelrySlot slot) {

\_name = name;

\_description = desc;

\_cost = cost;

\_isLore = lore;

\_isMagic = magic;

\_isTradeable = trade;

\_rarity = rare;

\_buffs = buffs;

\_slot = slot;

}

#endregion

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class DIALOGUESYSTEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace DialogueSystem {

public class Dialogue : IIdentity, IClonable<Dialogue>, IEditable {

private string \_name;

#region Getters & Setters

public string Name { get { return \_name; } set {\_name = value; } }

#endregion

#region Constructors

public Dialogue() {

\_name = “New Dialogue”;

}

public Dialogue Clone() {

return (Dialogue)MemberwiseClone();

}

#endregion

public void RevertFromDatabase() { }

public void OnInspectorGUI() { }

public void OnEditorGUI() { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM QuestState \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace QuestSystem {

public enum QuestState { Inactive, Active, Pending, Complete }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM QuestType \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace QuestSystem {

public enum QuestType { None, Collection, Deliver, Elimination, Interaction }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM ItemRarity \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public enum ItemRarity { Common, Uncommon, Rare, Epic, Legendary }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM EquipmentSlot \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public enum EquipmentSlot { NONE, HEAD, CHEST, LEGS, BELT, FEET, SHOULDERS, ARMS, HANDS, BACK }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM WeaponType \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public enum WeaponType { 1H\_SLASH,1H\_BLUNT,1H\_PIERCE, RANGED, 2H\_SLASH, 2H\_BLUNT, 2H\_PIERCE }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENUM JewelrySlot \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace ItemSystem {

public enum JewelrySlot { NONE, EARRING, FINGER, NECKLACE, CHARM }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class EVENTMANAGER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer.QuestSystem {

public delegate void SFEventHandler<T> (T sender);

public delegate void SFEventHandler<T, U> (T sender, U args);

public static class EventManager {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* interface IIdentity \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer.Source {

public interface IIdentity {

string Name { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* interface IClonable \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer.Source {

public interface IClonable<T> {

T Clone(T obj);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* interface IEditable \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer.Source {

public interface IEditable {

void OnInspectorGUI();

void OnEditorGUI();

void RevertFromDatabase();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* interface IPrefab \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer.Source {

public interface IPrefab {

private GameObject \_prefab;

public GameObject Prefab { get { return \_prefab; } set { \_prefab = value; } }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Buff \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class Buff {

private string buffName;

private int buffValue;

public Buff(string name, int value) {

buffName = name;

buffValue = value;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Test \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

using Spacefarer.QuestSystem;

namespace Spacefarer {

public class Test : MonoBehaviour {

private Quest \_quest;

private Item \_item;

private void Awake() {

\_item = new Item () { name = “Test Item” };

\_quest = new Quest (“Our first quest”, “Go to the wood”, QuestType.Collection, string.Empty, \_item.name, 5);

}

private void OnGUI() {

GUILayout.BeginArea (new Rect(20,20,Screen.width-40, Screen.Height-40));

GUILayout.Label(string.Format(“Name: \t{0}”, \_quest.name));

GUILayout.Label(string.Format(“Description: \t{0}”, \_quest.description));

GUILayout.Label(string.Format(“Type: \t{0}”, \_quest.type));

GUILayout.Label(string.Format(“Actor Name: \t{0}”, \_quest.actorName));

GUILayout.Label(string.Format(“Item Name: \t{0}”, \_quest.itemName));

GUILayout.Label(string.Format(“Quantity Needed: \t{0}”, \_quest.quantityNeeded));

GUILayout.Label(string.Format(“Quantity Remaining: \t{0}”, quest.quantityRemaining));

GUILayout.Label(string.Format(“Optional?: \t{0}”, \_quest.isOptional));

GUILayout.Label(string.Format(“Status:\t{0}”, \_quest.status));

GUILayout.Space(40.0f);

if (GUILayout.Button(“Begin Quest”)) {

\_quest.status = QuestStatus.Active;

}

if (GUILayout.Button(“Collect 1 ” + \_item.name)) {

Item.CollectItem(\_item);

}

if (GUILayout.Button(“Complete Quest”)) {

\_quest.Complete();

}

GUILayout.EndArea ();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class ActorContainer \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace Spacefarer {

public class ActorContainer : MonoBehaviour {

private Actor \_actor;

public actor { get { return \_actor; } }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class ActorEditor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEditor;

namespace Spacefarer.Editor {

[CustomEditor(typeof(ActorContainer))]

public class ActorEditor : Editor {

private Actor \_target;

public override void OnInspectorGUI() {

\_target = ((ActorContainer) target).actor;

if(\_target != null)

\_target.OnInspectorGUI();

else

if (GUILayout.Button(“New Actor”))

\_target = new Actor();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Database \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

using System.Collections.Generic;

using ItemSystem;

using QuestSystem;

using ActorSystem;

using DialogueSystem;

namespace Spacefarer {

public class Database : ScriptableObject {

public static readonly string database\_location = @”Assets/Data/Database.asset”;

public Dictionary <string, Actor> actors;

public Dictionary <string, Item> items;

public Dictionary <string, Quest> quests;

public Dictionary <string, Dialogue> dialogues;

public void AddActor(Actor actor) {

actors.Add(actor.name, actor);

}

public void AddItem(Item item) {

items.Add(item.name, item);

}

public void AddQuest(Quest quest) {

quests.Add(quest.name, quest);

}

public void AddDialogue(Dialogue dialogue) {

dialogues.Add(dialogue.name, dialogue);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class DatabaseEditor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEditor;

using UnityEngine;

namespace Spacefarer {

public class databaseEditor : EditorWindow {

private static readonly Vector2 \_window\_min\_size = new Vector2(800.0f, 600.0f);

private Database \_database;

[MenuItem(“Window/Database Editor”)]

public static void GetWindow() {

DatabaseEditor window = GetWindow<DatabaseEditor>(“Database Editor”, true);

window.minSize = window\_min\_size;

}

private void OnEnable() {

\_database = ScriptableObjectManager.LoadAsset<Database>(Database.database\_location);

}

private void OnGUI() {

EditorGUILayout.LabelField(“Database Exists: ”, \_database != null ? “True” : “False ”);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class ScriptableObjectManager \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

using UnityEditor;

namespace Spacefarer.Utility {

public static class ScriptableObjectManager {

public static void CreateAsset<T>() where T : ScriptableObject {

var asset = ScriptableObject.CreateInstance<T>();

ProjectWindowUtil.CreateAsset (asset, “New “ + typeof(T).Name + “.asset”);

}

public static void CreateAsset<T>(string fileLocation, string fileName) where T : ScriptableObject {

CreateAsset<T>(Path.Combine(fileLocation, filename + “.asset”);

}

public static void CreateAsset<T>(string path) where T : ScriptableObject {

T asset = ScriptableObject.CreateInstance<T>();

AssetDatabase.CreateAsset (asset, path);

AssetDatabase.SaveAssets ();

AssetDatabase.Refresh ();

}

public static T LoadDatabase<T>(string path) where T: ScriptableObject {

return AssetDatabase.LoadAssetAtPath<T>(path);

}

public static T LoadAsset<T>(string fileLocation, string fileName) where T : ScriptableObject {

return LoadAsset<T>(Path.Combine(fileLocation, filename + “.asset”);

}

[MenuItem(“Assets/Create/Database”)]

public static void CreateYourScriptableObject() {

CreateAsset<Database> ();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class Game \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using UnityEngine;

using System.IO;

namespace Spacefarer {

public static class Game {

private static Database \_database;

private static readonly string database\_location = Path.Combine(Application.streamingAssetsPath, “Data/Database.asset”);

public static void Initialize() {

\_database = LoadDatabase();

}

public static void CreateDatabase() {

ScriptableObjectManager.CreateAsset<Database>(database\_location);

}

public static Database LoadDatabase() {

return ScriptableObjectManager.LoadAsset<Database>(database\_location);

}

}

}